ST. XAVIER’S COLLEGE

**Maitighar, Kathmandu**

**(Affiliated to Tribhuvan University)**



**Database Management System**

Lab assignment #2

**Submitted By**

Biraj Poudyal

B.Sc. CSIT

Year II/IV Semester

013BSCIT015

**Submitted To**

Er. Sanjay Kumar Yadav

Lecturer

Department of Computer Science

St. Xavier’s College

Maitighar, Kathmandu

**Submitted On**

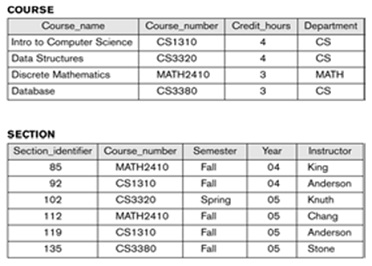
July 27, 2015

**Simplified database system:**

A **d**ata**b**ase **m**anagement **s**ystem (DBMS) is a collection of[programs](http://www.webopedia.com/TERM/P/program.html) that enables you to [store](http://www.webopedia.com/TERM/S/store.html), modify, and extract information from a [database](http://www.webopedia.com/TERM/D/database.html). There are many different types of database management systems, ranging from small [systems](http://www.webopedia.com/TERM/S/system.html) that [run](http://www.webopedia.com/TERM/R/run.html) on[personal computers](http://www.webopedia.com/TERM/P/personal_computer.html) to huge systems that run on [mainframes](http://www.webopedia.com/TERM/M/mainframe.html).

A DBMS (database management system) is a collection of programs that enables users to create and maintain database. The DBMS is a common purpose software system that facilitates the process of constructing, defining, manipulating and sharing databases among various users as well as applications. Defining a database state the database involves specifying the constraints, data types and structures of the data to be stored in the database. The descriptive information is as well stored in the database in the form database catalogue or dictionary- it is called meta-data. Manipulating the data comprises the querying the database to retrieve the specific data. An application program accesses the database through transferring the quarries or requests for data to DBMS.The significant function provided by the DBMS includes protecting the database and maintain the database.

***Example of a simple Database***:



***Example of a Student File***:



**2. Approaches to management of data**

* Database approach

In order to remove all limitations of the File Based Approach, a new approach was required that must be more effective known as Database approach.

The Database is a shared collection of logically related data, designed to meet the [information](http://ecomputernotes.com/fundamental/information-technology/what-do-you-mean-by-data-and-information) needs of an organization. A database is a [computer](http://ecomputernotes.com/fundamental/introduction-to-computer/what-is-computer) based record keeping system whose over all purpose is to record and maintains information. The database is a single, large repository of data, which can be used simultaneously by many departments and users. Instead of disconnected files with redundant data, all data items are integrated with a minimum amount of duplication.

The database is no longer owned by one department but is a shared corporate resource. The database holds not only the organization's operational data but also a description of this data. For this reason, a database is also defined as a self-describing collection of integrated records. The description of the data is known as the Data Dictionary or Meta Data (the 'data about data'). It is the self-describing nature of a database that provides program-data independence.

* **File system approach**

1. In traditional approach, information is stored in flat files which are maintained by the file system under the operating system’s control.
2. Application programs go through the file system in order to access these flat files

**How data is stored in flat files**

* Data is stored in flat files as records.
* Records consist of various fields which are delimited by a space, comma, pipe, any special character etc.
* End of records and end of files will be marked using any predetermined character set or special characters in order to identify them

# Some advantages of using the database management system approach rather than the file system approach

A database is just an abstraction layer on top of the file system that provides the client with a generic framework for dealing with storage, concurrency, consistency, etc. Its analogous to the way that a higher level programming language like Java provides an abstraction that makes it easier for a programmer to write an application than would otherwise be the case if she were writing in machine language by hand.  
  
A database is generally a good fit if you are doing anything that involves information with various parts which can change separately during concurrent access. Most database systems will provide access control, transactions, referential integrity, indexing and other tools to make your life easier. If you stored this kind of data directly on the filesystem, you'd have to manually build a lot of logic in your application to deal with these sorts of issues or risk data inconsistency.

1. **Database V/S file system approaches**

This topic has been on my plate for some time now. It's interesting to see how databases have come a long way and have clearly out-shadowed file-systems for storing structured or unstructured information.

Technically, both of them support the basic features necessary for data access. For example, both of them:

* Ensure data is managed to ensure its integrity and quality
* Allow shared access by a community of users
* Use well-defined schema for data-access
* Support a query language

But file-systems seriously lack some of the critical features necessary for managing data. Lets take a look at some of these features.

* **Data abstraction**

For the system to be usable, it must retrieve data efficiently. The need for efficiency has led designers to use complex data structures to represent data in the database. Since many database-systems users are not computer trained, developers hide the complexity from users through several levels of abstraction, to simplify users’ interactions with the system. The three levels of data abstraction:

* **Physical Level** : The lowest level of abstraction describes how the data are actually stored. The physical level describes complex low-level data structures in detail.
* **Logical Level** : The next-higher level of abstraction describes what data are stored in the database, and what relationships exist among those data. The logical level thus describes the entire database in terms of a small number of relatively simple structures. Although implementation of the simple structures at the logical level may involve complex physical-level structures, the user of the logical level does not need to be aware of this complexity. Database administrators, who must decide what information to keep in the database, use the logical level of abstraction.
* **View Level** : The highest level of abstraction describes only part of the entire database. Even though the logical level uses simpler structures, complexity remains because of the variety of information stored in a large database. Many users of the database system do not need all this information; instead, they need to access only a part of the database. The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database.
* **Reliability**

It manages the reliability of read/write requests by interacting with the buffer manager It may generate new read/write requests for reliability purposes .It exploits the log file a persistent archive recording DBMS activity stored on stable memory It prepares data for performing recovery by means of the operations checkpoint dump.

* **Efficiency /performance**

In this section we discuss some of the more "general" methods that can be used for database performance evaluation. The word "general" is binding to systems, meaning that the approaches mentioned here are generally true for "systems" with a special focus on database systems.

According to , performance analysis of database systems serve two basic purposes:

1. For the evaluation of the best configuration and operating environment of a single database system, and

2. Studying two or more database systems and providing a systematic comparision of the systems.

Accordingly, some of the analytical modeling methods for evaluating systems that are applicable for database systems too are:

1. Queuing Models: Queuing models are effective to study the dynamics of a database system when it is modeled as a multi-component system with resource allocation constraints and jobs moving around from one component to another. Examples of such dynamic studies are concurrent transaction control algorithms, data allocation and management in distributed database systems etc.

2. Cost Models: Cost Models are useful in studying the cost in terms of Physical storage and query processing time. The cost model gives some real insight into the actual physical structure and performance of a database system.

3. Simulation Modeling: A simulation Modeling is more effective for obtaining better estimates since it not only analyses the database system in isolation but can effectively analyze the database system with the application program running on top of it and the database system itself operating within the constrained environment of an operating system on a real physical hardware.

4. Benchmarking: Benchmarking is the best method when multiple database systems need to be evaluated against each other but suffer from the inherent setback that it assumes all systems to be fully installed and operational. Benchmarking relies on the effectiveness of the synthetic workloads. Real workloads are non repeatable and hence not good for effective benchmarking.

presents a 3 step process for benchmark evaluation

a. Benchmark design

b. Benchmark Execution

c. Benchmark analysis

Our study makes a delves deeper into the various database system evaluation benchmarks and we leave the more detailed analysis of Database systems benchmark studies to a later section where we see different real-life benchmarking techniques.

proposes a set of methods for database performance evaluation assuming the system to be operating in a multi-user environment. Accordingly the three factors that effect the performance of a database system in a multi-user environment are:

1. Multi-programming level

2. Query Mix

3. Extent of data Sharing

Data sharing is the condition of concurrent access of a data object by multiple processes. The interesting factor here is that of the query mix. A proper query mix needs to test the appropriate levels of CPU and disk utilization required to serve a particular query. The query mix needs to properly represent a true multi-user environment. Also, the query mix may be designed to represent a certain percentage of data sharing. Once these have been figured out, the query-mix forms a representative benchmark program and multiple copies of the bench-mark program are issued concurrently to simulated multi-programming effects. Also, different query-mixes allow diversity in the experimental design conditions. The response variable studied is system throughput and response time.

Summarizing, in this section, the focus was primarily to study the performance evaluation techniques considering the "general system criterion" of a database system. In the next section we look at the performance evaluation techniques more specialized for particular database system types.

1. **Three layer architecture**

Three-tier architecture allows any one of the three tiers to be upgraded or replaced independently. The user interface is implemented on a desktop PC and uses a standard graphical user interface with different modules running on the application server. The relational database management system on the database server contains the computer data storage logic. The middle tiers are usually multitiered.

The three tiers in a three-tier architecture are:

1. Presentation Tier: Occupies the top level and displays information related to services available on a website. This tier communicates with other tiers by sending results to the browser and other tiers in the network.
2. Application Tier: Also called the middle tier, logic tier, business logic or logic tier, this tier is pulled from the presentation tier. It controls application functionality by performing detailed processing.
3. Data Tier: Houses database servers where information is stored and retrieved. Data in this tier is kept independent of application servers or business logic.
4. **Advantage and disadvantage of DBMS**

**Advantage of DBMS :**  
The DBMS serves as the intermediary between the user and the database. The database structure itself is stored as a collection of files, and the only way to access the data in those files is through the DBMS. The DBMS receives all application requests and translates them into the complex operations required to fulfill those requests. The DBMS hides much of the database’s internal complexity from the application programs and users.  
**The different advantages of DBMS are as follows.**  
1. **Improved data sharing.**  
The DBMS helps create an environment in which end users have better access to more and better-managed data. Such access makes it possible for end users to respond quickly to changes in their environment.  
**2. Improved data security.**  
The more users access the data, the greater the risks of data security breaches. Corporations invest considerable amounts of time, effort, and money to ensure that corporate data are used properly. A DBMS provides a framework for better enforcement of data privacy and security policies.  
**3. Better data integration.**  
 Wider access to well-managed data promotes an integrated view of the organization’s operations and a clearer view of the big picture. It becomes much easier to see how actions in one segment of the company affect other segments.  
**4. Minimized data inconsistency.**  
 Data inconsistency exists when different versions of the same data appear in different places. For example, data inconsistency exists when a company’s sales department stores a sales representative’s name as “Bill Brown” and the company’s personnel department stores that same person’s name as “William G. Brown,” or when the company’s regional sales office shows the price of a product as $45.95 and its national sales office shows the same product’s price as $43.95. The probability of data inconsistency is greatly reduced in a properly designed database.  
**5. Improved data access.**  
The DBMS makes it possible to produce quick answers to ad hoc queries. From a database perspective, a query is a specific request issued to the DBMS for data manipulation—for example, to read or update the data. Simply put, a query is a question, and an ad hoc query is a spur-of-the-moment question. The DBMS sends back an answer (called the query result set) to the application. For example, end users, when dealing with large amounts of sales data, might want quick answers to questions (ad hoc queries) such as:  
- What was the dollar volume of sales by product during the past six months?  
- What is the sales bonus figure for each of our salespeople during the past three months?  
- How many of our customers have credit balances of $3,000 or more?  
**6.Improved decision making.**  
Better-managed data and improved data access make it possible to generate better-quality information, on which better decisions are based. The quality of the information generated depends on the quality of the underlying data. Data quality is a comprehensive approach to promoting the accuracy, validity, and timeliness of the data. While the DBMS does not guarantee data quality, it provides a framework to facilitate data quality initiatives.  
**7.Increased end-user productivity.**  
The availability of data, combined with the tools that transform data into usable information, empowers end users to make quick, informed decisions that can make the difference between success and failure in the global economy.  
  
**Disadvantages of Database:**  
  
Although the database system yields considerable advantages over previous data management approaches, database systems do carry significant disadvantages. For example:  
**1. Increased costs.**  
Database systems require sophisticated hardware and software and highly skilled personnel. The cost of maintaining the hardware, software, and personnel required to operate and manage a database system can be substantial. Training, licensing, and regulation compliance costs are often overlooked when database systems are implemented.  
**2. Management complexity.**  
Database systems interface with many different technologies and have a significant impact on a company’s resources and culture. The changes introduced by the adoption of a database system must be properly managed to ensure that they help advance the company’s objectives. Given the fact that database systems hold crucial company data that are accessed from multiple sources, security issues must be assessed constantly.  
**3. Maintaining currency.**  
To maximize the efficiency of the database system, you must keep your system current. Therefore, you must perform frequent updates and apply the latest patches and security measures to all components. Because database technology advances rapidly, personnel training costs tend to be significant. Vendor dependence. Given the heavy investment in technology and personnel training, companies might be reluctant to change database vendors. As a consequence, vendors are less likely to offer pricing point advantages to existing customers, and those customers might be limited in their choice of database system components.  
**4. Frequent upgrade/replacement cycles.**  
DBMS vendors frequently upgrade their products by adding new functionality. Such new features often come bundled in new upgrade versions of the software. Some of these versions require hardware upgrades. Not only do the upgrades themselves cost money, but it also costs money to train database users and administrators to properly use and manage the new features.

**6. Drawback of using file system approach:**

1. Data Security

The data stored in the flat file(s) can be easily accessible and hence it is not secure.

Example: Consider an online banking application where we store the account related information of all customers in flat files. A customer will have access only to his account related details. However from a flat file, it is difficult to put such constraints. It is a big security issue.

1. Data Redundancy

In this storage model, the same information may get duplicated in two or more files. This may lead to to higher storage and access cost. it also may lead to data inconsistency.

For Example, assume the same data is repeated in two or more files. If a change is made to data stored in one file, other files also needs to be change accordingly.

Example: Assume employee details such as firstname, lastname, emailid are stored in employee\_details file and employee\_salary file. If a change needs to be made to emailid, both employee\_details file and emplyee\_salary file need to be updated otherwise it will lead to inconsistent data.

However, it is possible to design file systems with minimal redundancy. Also note that Data redundancy is sometimes preferred.

Example: Assume employee details such as firstname, lastname, emailid are stored only in employee\_details file and not in employee\_salary file. If we need to access an employee salary along with firstname of the employee, we have to retrieve details from two files. This would mean an increased overhead.

1. Data Isolation

Data Isolation means that all the related data is not available in one file. Usually the data is scattered in various files having different formats. Hence writing new application programs to retrieve the appropriate data is difficult.

1. Program/Data Dependence

In traditional file approach, application programs are closely dependent on the files in which data is stored. If we make any changes in the physical format of the file(s), like addition of a data field , etc, all application programs needs to be changed accordingly. Consequently, for each of the application programs that a programmer writes or maintains, the programmer must be concerned with data management. There is no centralized execution of the data management functions. Data management is scattered among all the application programs.

Example: Consider the banking system. An employee\_salary file exists which has details about the salary of employees. An employee\_salary record is described by

employee\_id

firstname

lastname

salary\_amount

An application program is available to display all the details about the salary of all employees. Assume a new data field, the date\_of\_joining is added to the employee\_salary file. Since the application program depends on the file, it also needs to be altered.

If the physical format of the employee\_salary file for example the field delimiter, record delimiter, etc. are changed, it necessitates that the application program which depends on it, also be altered.

1. Lack of Flexibility

The traditional systems are able to retrieve information for predetermined requests for data. If we need unanticipated data, huge programming effort is needed to make the information available, provided the information is there in the files. By the time the information is made available, it may no longer be required or useful.

Example : Consider a software application which is able to generate employee salary report. Assume that all the data is stored in flat files. Suppose we now have a requirement to retrieve all the employee details whose salary is greater than Rs.10000. It is not easy to generate such on-demand reports and lot of time is needed for application developers to modify the application to meet such requirements.

1. Concurrent Access Anomalies

Many traditional systems allow multiple users to access and update the same piece of data simultaneously. However this concurrent updates may result in inconsistent data. To guard against this possibility, the system must maintain some form of supervision. But supervision is difficult because data may be accessed by many different application programs and these application programs may not have been coordinated previously.

Example : Consider a personal information system which has the data of all employees. Now there may be an employee updating his address details in the system and at the same time, an administrator may be taking a report containing the data of all employees. This is called concurrent access. Since the employee's address is being updated at the same time, there is a possibility of the administrator reading an incorrect address.

These difficulties lead to the development of database systems.

1. **References**

* [**http://database-management-systems.blogspot.com/2009/09/data-abstraction.html**](http://database-management-systems.blogspot.com/2009/09/data-abstraction.html)
* [**http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter1/node4.html**](http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter1/node4.html)
* [**https://cs.uwaterloo.ca/~tozsu/courses/cs448/notes/Reliability.pdf**](https://cs.uwaterloo.ca/~tozsu/courses/cs448/notes/Reliability.pdf)
* [**http://www.myreadingroom.co.in/about/226-advantages-and-disadvantages-of-dbms.html**](http://www.myreadingroom.co.in/about/226-advantages-and-disadvantages-of-dbms.html)
* [**http://www.cse.wustl.edu/~jain/cse567-08/ftp/db/#Section3**](http://www.cse.wustl.edu/~jain/cse567-08/ftp/db/#Section3)**.**
* Yao-S Bing, Alan R. Hevner, A Guide to Performance Evaluation of Database Systems, Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402,1984
* Haran Boral, David J DeWitt, A methodology for database system performance evaluation, ACM SIGMOD Record, Volume 14, Issue 2, June 1984